جامعة دمشق

كلية الهندسة المعلوماتية

اختصاص برمجيات

السنة الخامسة

مشروع مادة النظم الموزعّة

مشرف المادة: المهندسة آية المعطي

إعداد الطلاب

روان محمّد إبراهيم

زهراء عدنان الصّوص

سرى خالد أبو راس

جمال أحمد المرشد

مقدمة عن RMI

(Remote Method Invocation)

هي تقنية تستخدم في لغة البرمجة جافا لتمكين تنفيذ أساليب (Methods) كائنية التوجه (Object-oriented methods)على كائنات (Objects) موجودة في آلة افتراضية جافا (JVM)أخرى. بعبارة أخرى، تتيح RMI التواصل بين الكائنات التي تعمل على أجهزة حاسوب مختلفة عبر الشبكة.

1. **الأساسيات:**
- **العميل :(Client)** هو التطبيق الذي يستدعي الأساليب على الكائن البعيد.
- **الخادم :(Server)** هو التطبيق الذي يحتوي على الكائنات البعيدة التي يتم استدعاء الأساليب عليها.
- **الكائن البعيد :(Remote Object)** هو الكائن الذي يمكن استدعاء أساليبه عن بُعد.

2. **آلية العمل:**
- يقوم الخادم بإنشاء كائنات بعيدة وتسجيلها مع مخدم التسجيل.(Registry)
- يقوم العميل بالحصول على مرجع لهذه الكائنات من مخدم التسجيل.
- عندما يستدعي العميل أسلوبًا على الكائن البعيد، يتم تنفيذ هذا الأسلوب على الخادم، ويتم إرسال النتائج إلى العميل.

3. **المكونات الرئيسية:**
- واجهة الكائن البعيد :(Remote Interface) تحدد الأساليب التي يمكن استدعاؤها عن بُعد.
- تنفيذ الكائن البعيد :(Remote Object Implementation) يوفر تنفيذًا فعليًا لواجهة الكائن البعيد.
- العميل :(Client) يستدعي الأساليب المحددة في الواجهة البعيدة.
- مخدم التسجيل :(Registry Server) يسجل الكائنات البعيدة بحيث يمكن للعميل العثور عليها.

4. **العملية العامة:**
- التسجيل: يقوم الخادم بإنشاء كائنات بعيدة وتسجيلها باستخدام Naming.rebind() أو Naming.bind().
- البحث: يستخدم العميل Naming.lookup() للعثور على الكائن البعيد.
- استدعاء الأسلوب: يستدعي العميل الأساليب على الكائن البعيد كما لو كان محليًا.

شرح التطبيق:

في تطبيقنا يمكن تحديد الخدمات البعيدة بتصوير لقطة شاشة او صورة كاميرا والتسجيل وهي خاصة بالموظف

اما بالنسبة للمدير فهو سيقوم باسترجاع بيانات جميع الموظفين حاليا او استرجاع بينات موظف حسب ال id.

قمنا ببناء الواجهة Interface التي تحوي تعريف الدوال التي ذكرت بالإضافة الى مصفوفة تمثل الموظفين النشطين

عند اخذ لقطة شاشة او صورة كاميرا سيتم حفظها في مجلد مؤقت بشكل تلقائي كي يراها الآدمن واذا أراد حفظها بشكل دائم ستحفظ في مجلد اخر سنشرح هذا لاحقا

بعد تعريف الواجهة البعيدة في RMI يجب تعريف الصف الذي يحوي تحقيق دوال هذه الواجهة وهو الصف **RmiInterfaceImplementation**

وقبل شرحه لقد قمنا بتعريف صف يمثل الموظف يحوي رقمه و ip الجهاز و غرض rmi الخاص به وعند تسجيل الموظف كما سنرى لاحقا سيتم تسجيل غرض من الصف employee, فاذا أراد المدير اخذ لقطة شاشة سيقوم فقط باستدعاء الدالة من غرض rmi من ال employee ويحوي دالة اخذ صورة او لقطة شاشة التي تستدعي الدالة من الواجهة البعيدة لأخذ صورة او لقطة شاشة ومن ثم حفظها في مجلد الصور المؤقتة

شرح RmiInterfaceImplementation

الدالة RegisterEmployeeData:

تحقيق الدالة التي يقوم المستخدم باستدعائها من اجل التسجيل في السيرفر وفيها يتم فحص id المستخدم إذا كان مسجل بشكل مسبق يتم حذفه ومن ثم إضافته والا فقط يتم اضافته

الدالة getEmployee:

يتم استدعاؤها من قبل مدير النظام لعرض بيانات موظف نشط حيث يمرر لها رقم الموظف وتقوم هذه الدالة بالمرور على مصفوفة الموظفين ومن ثم اختبار فيما اذا كان ال id المدخل موجود ام لا في حال كان موجود تعيد بيانات الموظف والا تعيد Null

الدالة getEmployees

تعيد مصفوفة الموظفين لدى المدير من اجل عرض بياناتهم

الدالة takeScreenshot:

تقوم هذه الدالة باستخدام الصف Robot من اجل اخذ لقطة شاشة

الدالة takeCameraPicture:

تقوم هذه الدالة باستخدام الصف Webcam من اجل اخذ صورة من الكاميرا

ملاحظة:

في كلا الدالتين نحول الصورة لمصفوفة بايتات وذلك لإرسالها عبر الشبكة حيث يعتبر تطبيق RMI شبكي حتى لو كان يعمل على نفس الجهاز وقمنا بتحويل الصورة الى مصفوفة بايتات من خلال استدعاء الدالة FormatImageForSend من الصف ImageOperation .

الصف ImageOperation:

يحوي هذا الصف على دالتين الأولى لتحويل الصورة الى مصفوفة بايتات والثانية لتحويل مصفوفة البايتات الى صورة من اجل حفظها كما سنرى لاحقا, ويحوي أيضا دالة لحفظ الصورة بشكل مؤقت **SaveTempImage** عند طلبها من قبل مدير النظام في القرص D في المجلد temp لذلك يجب انشاء المجلد في القرص d قبل تشغيل المشروع , كما يحوي الصف على دالة **SaveHistoricalImage** لحفظ الصورة بشكل دائم في مجلد historical لذلك يجب انشاء مجلد على القرص d بنفس الاسم قبل تشغيل المشروع , حيث يمرر لهذه الدالة الصورة والاسم الذي يمثل id والنوع لقطة شاشة او صورة كاميرا ويتم في الحفظ إضافة timestamp لتجنب حفظ صورتين بنفس الاسم.

الصف RmiRemoteServer:

هو الكود الخاص بالسيرفر البعيد المسؤول عن تسجيل الخدمة

وفيه يتم تعريف غرض من الصف RmiInterfaceImplementation ومن ثم تسجيله على البورت 5000 واسم الخدمة rmi

الصف EmployeeHost:

بشكل عام سيقوم التطبيق بالاتصال بالسيرفر البعيد وانشاء غرض من الصف Employee وتسجيله في النظام كل فترة زمنية ومن ثم فتح socket لاستقبال رسائل الأدمن وعرضها

الدالة main:

سيتم استدعاء دالة تم تعريفها للاتصال بالسيرفر وتعيد غرض من الواجهة البعيد ومن ثم تقوم بإنشاء غرض Employee تمرر له ip الجهاز وغرض الواجهة البعيدة ورقم البورت الذي سينتظر الرسائل عليه وهو 5001 ومن ثم يقوم باستدعاء الدالة المسؤولة عن تسجيل الموظف ويمرر لها غرض الواجهة البعيدة stub وغرض بيانات الموظف ومن ثم يتم استدعاء الدالة المسؤولة عن انتظار رسائل مدير النظام

الدالة ConnectToRMI:

وهي الدالة المسؤولة عن الاتصال بالمخدم البعيد للحصول على واجهة rmi للاستفادة من الخدمات ويتم الاتصال عنفس اسم الخدمة المحدد rmi والبورت 5000

الدالة GetClientHostIP:

تعيد ip الجهاز وفي حالتنا ستعيد localhost بشكل افتراضي

الدالة registerToRemoteServer:

وهي الدالة المسؤولة عن التسجيل في المخدم البعيد وفيها قمنا بإنشاء مسلك لجعل عملية التسجيل تعمل بشكل متواز مع بقية العمليات وليس بشكل تسلسلي كي لا يتوقف الكود عند التسجيل لان عملية التسجيل سيتم استدعاؤها كل دقيقتين

120000 ملي ثانية وضمن المسلك تم تعريف مؤقت Timer وفيه يتم استدعاء الدالة البعيدة intf.RegisterEmployeeData(emp) المسؤولة عن تسجيل الموظف فعليا

الدالة WaitAdminMessage:

ويتم فيها انتظار رسائل المدير حيث يتم تعريف serverSocket لانتظار اتصال من المدير ومن ثم يتم تعريف socket ومسلك لاستلام الرسائل بشكل منفصل وعرضها على الشاشة فور استلامها.

وبذلك نكون قد أنهينا الصف الخاص بالزبون نلاحظ ان الزبون لم يستدعي سوى الدالة البعيدة المسؤولة عن التسجيل ودوال لقطة الشاشة سيتم استدعاؤها من قبل مدير النظام

الصف AdminHost:

هو الصف الذي يمثل مدير النظام حيث انه في البداية سيتصل مع المخدم البعيد ومن ثم سيعرض له التطبيق الموظفين النشطين باستدعاء الدالة البعيدة التي تعيد بيانات جميع الموظفين النشطين ومن ثم يطلب منه ادخال رقم الموظف لإجراء العمليات وعند ادخال رقم موظف يتم استدعاء الدالة البعيدة التي تعيد بيانات الموظف حسب رقمه فاذا طلب المدير اخذ لقطة شاشة يتم استدعاء الدالة من الغرض e الذي يمثل الموظف لأخذ لقطة شاشة ومن ثم يظهر خيار الحفظ لحفظ الصورة في مجلد الصور الدائمة وكذلك الامر بالنسبة لصورة الكاميرا واذا طلب إرسال رسالة يتم استدعاء الدالة sendMessage التي تقوم بالاتصال بجهاز الموظف عن طريق ip و البورت ومن ثم ارسال الرسالة له.

السؤال الثاني

1-تم إنشاء 4 خدمات لهذا النظام

Gateway Service

- تعمل كبوابة وحيدة للدخول إلى جميع الخدمات في النظام.
- توجيه الطلبات إلى الخدمات المناسبة بناءً على المسارات (Routes) المحددة.

Discovery Service

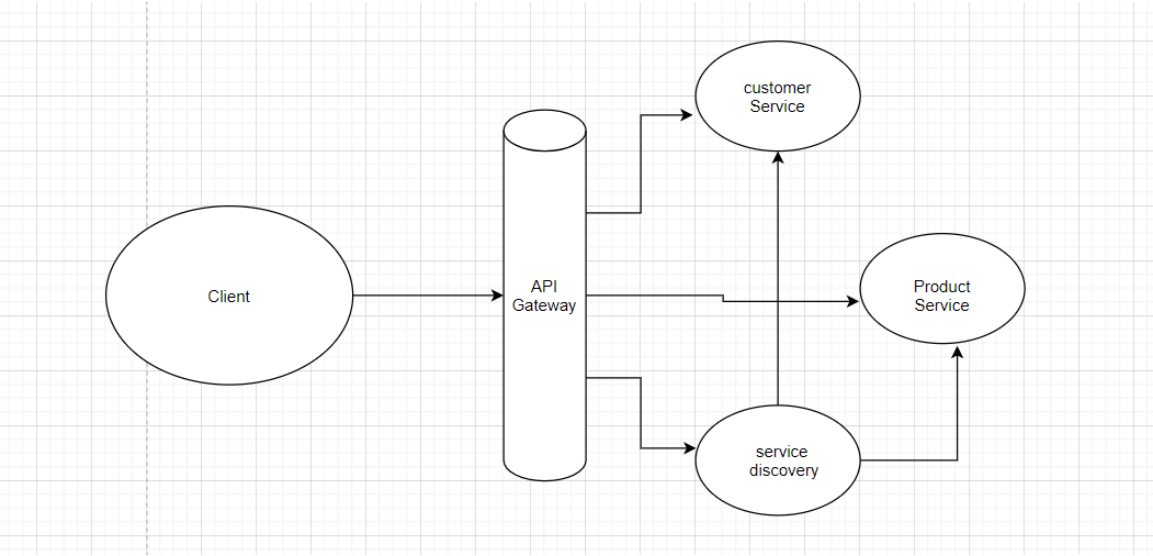
- تتيح تسجيل واكتشاف الخدمات المتاحة بشكل ديناميكي.
- تستخدم لتحديد مواقع الخدمات النشطة وضمان التوزيع الديناميكي للطلبات.

Customer service

- إدارة معلومات العملاء: إضافة، تعديل، حذف، واستعراض العملاء.
- شراء منتجات من قبل العملاء

Product Service

- إدارة معلومات المنتجات: إضافة، تعديل، حذف، واستعراض المنتجات.



2 – تم استخدام ال structure التالية Controller – entitie – repository – service

3- انشاء خدمة ال gateway وربطها مع customer و product و service discovery

```
spring.application.name=gateway

server.port=8080

spring.cloud.gateway.routes[0].id=productService
spring.cloud.gateway.routes[0].uri=lb://productService
spring.cloud.gateway.routes[0].predicates[0]=Path=/product/


spring.cloud.gateway.routes[1].id=customerService
spring.cloud.gateway.routes[1].uri=lb://customerService
spring.cloud.gateway.routes[1].predicates[0]=Path=/customer/




eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/
eureka.instance.hostname=localhost
```

4-ربط خدمة customer service مع service discovery

```
spring.application.name=customerService
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url = jdbc:mysql://localhost:3306/customerService
spring.datasource.username = root
spring.datasource.password =
spring.jpa.show-sql = true
spring.jpa.hibernate.ddl-auto = update
server.port=8081


eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/
eureka.instance.hostname=localhost
```

ربط خدمة product service مع service discovery

```
spring.application.name=productService
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url = jdbc:mysql://localhost:3306/productservice
spring.datasource.username = root
spring.datasource.password =
spring.jpa.show-sql = true
spring.jpa.hibernate.ddl-auto = update
server.port=8082


eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/
eureka.instance.hostname=localhost
```

-5

تم ربط خدمة المنتج مع الزبون من اجل احضار المنتجات الخاصة بزبون معين باستخدام restTemplate

```
1 usage
public ResponseEntity<?> fetchCustomerById(int id){

    Optional<Customer> customers =  customerRepository.findById(id);

    if(customers.isPresent()){
        List<Product> product = restTemplate.exchange(
                url: "http://productService/product/by-customer/"+customers.get().getId(
                HttpMethod.GET,
                requestEntity: null,
                new ParameterizedTypeReference<List<Product>>() {}
        ).getBody();
        CustomerResponse customerResponse = new CustomerResponse(
                customers.get().getId(),
                customers.get().getName(),
                customers.get().getGender(),
                customers.get().getAge(),
                product
        );
        return new ResponseEntity<>(customerResponse, HttpStatus.OK);
    }else{
        return new ResponseEntity<>( body: "No Customer Found",HttpStatus.NOT_FOUND);
    }    }
```

6- دارة الفشل والتأخر في التواصل بين الخدمات

```java
@CircuitBreaker(name = "CustomerService",fallbackMethod = "fallBackGetProducts")
@Retry(name = "CustomerService")
public List<Product> getProducts(int customerId){
return    restTemplate.exchange(
            url: "http://productService/product/by-customer/"+customerId,
            HttpMethod.GET,
            requestEntity: null,
            new ParameterizedTypeReference<List<Product>>() {}
    ).getBody();
}
no usages
public List<Product> fallBackGetProducts(){

    final Product product1 =new Product();
    List<Product> products = new ArrayList<>();
    products.add(product1);
    return   products;
}
```

7- من اجل توزيع الحمل تم استخدام annotation التالية

```java
@Bean
@LoadBalanced
public RestTemplate restTemplate() { return new RestTemplate(); }
```